Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

Fast and Memory Efficient Graph Optimization via ICM for Visual Place Recognition

Stefan Schubert
stefan.schubert@etit.tu-chemnitz.de

Peer Neubert
peer.neubert@etit.tu-chemnitz.de
Chemnitz University of Technology

Peter Protzel
peter.protzel@etit.tu-chemnitz.de

Abstract—**Visual place recognition is the task of finding same places in a set of database images for a given set of query images. This task becomes particularly challenging if the environmental condition changes between database and query, for example from day to night. In this paper, we build upon our recent work on graph optimization for place recognition, where a graph was used to model additional structural knowledge like sequences. A subsequent non-linear least squares optimization (NLSQ) improved the place recognition performance. While this approach achieves very high performance, it is quite slow and memory inefficient. This paper addresses the long runtime and the high memory usage in order to obtain the same or better place recognition performance faster on larger problems. We propose a novel graph optimization procedure that is based on Iterated Conditional Modes (ICM). In addition, we investigate a new cost function for an edge in the graph. Our novel ICM-based approach achieves 9.1msec *maximum runtime* per query, which is $260\times$ faster than the *minimum runtime* with NLSQ. Moreover, with ICM we can optimize problems that are not feasible with NLSQ on a full graph due to memory limitations. To demonstrate the superior performance of our ICM-based method, we provide extensive experimental evaluations with the essence of 987 precision-recall curves: Our proposed ICM-based method is compared to the NLSQ-based method as well as to six sequence-based approaches from the literature on 21 sequence combinations from five datasets with four different image descriptors. Our experiments show that our ICM-based method with sequence-exploitation not only improves the NLSQ-based performance by 10% on average while being $385\times$ faster and using more than $60\times$ less memory. It also significantly outperforms all six sequence-based methods from the literature by *at least* 32% on average with the NetVLAD descriptor while using comparable runtime and memory. Code is available online[1].**

I. Introduction

Visual place recognition is the task of finding same places in a set DB of M database images for a set Q of N query images. This becomes particularly challenging if the environmental condition changes between both sets, for example from day to night. Visual place recognition is required for tasks like loop closure detection in SLAM and candidate selection for global localization. It is subject of active research and many approaches for performance improvements have been proposed. These address different elements of the basic place recognition pipeline as shown in Fig. 1.

This work was supported by the German Federal Ministry for Economic Affairs and Energy.

[1]Source code: https://www.tu-chemnitz.de/etit/proaut/prstructure

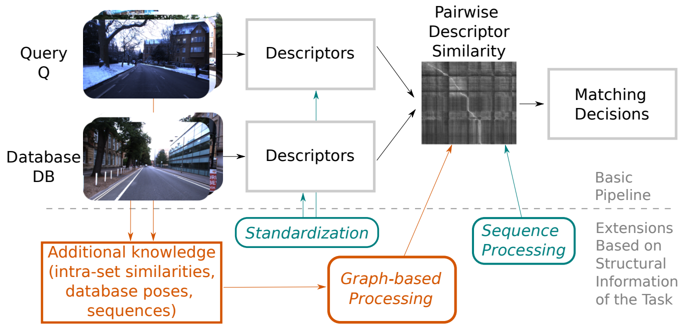


Fig. 1. The basic place recognition pipeline (above the horizontal dashed line) can be extended with additional information (below this line). Established approaches are the standardization of descriptors [31] and sequence processing (e.g., [22, 25]). We propose a novel optimization procedure for graph-based place recognition that fuses various sources of additional information in a single graph.

Many approaches exploit additional structural knowledge like spatio-temporal sequences in DB and Q, intra-database and intra-query similarities, or odometry. However, most methods focus on only one type of additional knowledge, even if more information would be available or use them successively rather than in combination. To address this problem, we proposed in our recent work [32] a graph-based framework for place recognition that fuses additional structural knowledge from different sources in a single graph. A subsequent optimization of the graph leads to significantly improved place recognition results. The experimental evaluation in [32] demonstrated a very good place recognition performance of the graph-based approach, which significantly outperformed multiple sequence-based methods from the literature.

A drawback of our graph-based method from [32] is the relatively long runtime and memory inefficiency: In our experiments, we measured a maximum runtime of $5.3\,\mathrm{sec}$ per query image and had to use partial optimizations of the graph as in [32] due to memory limitations. The reason for the long runtime and memory inefficiency is the used non-linear least squares optimization (NLSQ) of the graph. It has to operate on a sparse but potentially huge Jacobian matrix J and on an often very long vector of residuals r.

In this paper, we adopt our graph-based approach from [32] with its superior place recognition performance. We contribute

- a novel graph optimization procedure based on Iterated Conditional Modes (ICM) that significantly improves the runtime and memory efficiency (Sec. VI)

- a new cost function for an edge in the graph (Sec. IV)
- the sequence-based method SeqConv which is used during optimization with ICM (Sec. VII)
- an extensive experimental evaluation of our approach with comparisons to [32] and six sequence-based methods from the literature on four different descriptors and 21 datasets (Sec. VIII)

Further, we provide a short introduction to the graph-based approach from [32] (Sec. III), discuss why the NLSQ-based optimization is slow and memory inefficient (Sec. V) and conclude our work in Sec. IX.

II. Related Work

Visual place recognition in changing environments is a subject of active research. Challenges and properties of visual place recognition are discussed in [30]. An overview of existing methods is given in a survey from 2016 [19]. In this paper, we extend our recent graph-based approach for place recognition from [32]. Our method optimizes the pairwise image similarities $\hat{s}_{ij} \in \hat{S}$ from pairwise image descriptor comparisons between the database and query set. There are various image descriptors in the literature as discussed in Sec. II-A. Before optimization the used graph models dependencies between pairwise image similarities $\hat{s}_{ij}$ in $\hat{S}$ based on structural knowledge from intra-database similarities $\hat{S}^{DB}$, intra-query similarities $\hat{S}^{Q}$ (Sec. II-B) and spatio-temporal sequences in the database and query set (Sec. II-C). In [32] the graphical model is optimized via non-linear least squares optimization. In this paper, we propose the usage of the different optimization scheme ICM (Sec. II-D).

A. *Image descriptors for visual place recognition in changing environments*

As hand-designed local image feature detectors like SURF [5] fail under appearance changes [37, 10, 36], deep-learned feature detectors and descriptors like DELF [28] and D2-Net [9] have been proposed. While these local descriptors achieve comparatively high performance, they are relatively slow to compare. Therefore, holistic image descriptors have been proposed that express each image in a single vector and allow a fast descriptor comparison. Sünderhauf et al. [35] demonstrated that flattened intermediate CNN-layers like the *conv3*-layer from AlexNet [18] trained for image classification can be used as holistic image descriptor to match places despite appearance changes. HybridNet [7] uses a similar architecture with a flattened intermediate layer-output, but was additionally trained for place recognition to achieve higher performance. NetVLAD [2] is a deep-learned architecture that combines CNN-layer outputs with a trainable version of VLAD (vector of locally aggregated descriptors) [16]. The usage of VLAD [16] allows the computation of potentially more viewpoint robust descriptors. DenseVLAD [36] extracts hand-designed RootSIFT [1] features densely all over the image to circumvent the feature detection and combines these with VLAD. In [24], hyperdimensional computing was used to convert a set of local image descriptors into a holistic descriptor. The performance of holistic descriptors can be further improved by descriptor standardization [31].

B. *Intra-database and intra-query similarities for performance improvements*

The approach in this paper models dependencies between image similarities $\hat{s}_{ij} \in \hat{S}$ in a graph based on prior structural knowledge from intra-database similarities $\hat{S}^{DB}$, intra-query similarities $\hat{S}^{Q}$, and spatio-temporal sequences. In [33] intra-database similarities $\hat{S}^{DB}$ were used for a selection of matching candidates from the database to reduce the number of required image comparisons for efficient place recognition. Similarly, Vysotska et al. [40] exploited noisy GPS priors from the database and query set to identify matching candidates for a reduced number of image comparisons and for performance improvements by avoiding image comparisons between similar looking but distant places. In [26] intra-database similarities $\hat{S}^{DB}$ and intra-query similarities $\hat{S}^{Q}$ were used to resolve matching inconsistencies in $\hat{S}$ for performance improvements.

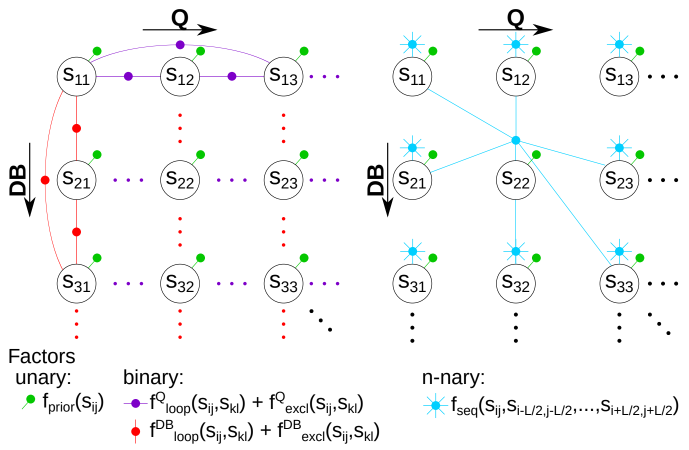
C. *Sequence-based methods for place recognition*

As explained in Sec. III-A, sequence information can be used if both the database and the query set are recorded as spatio-temporal sequences. There is a wide range of existing sequence-based methods for place recognition in the literature like our graph-based approach in this paper that exploit this type of information. Most methods operate on the image similarities $\hat{s}_{ij} \in \hat{S}$ to refine their values. [23] uses a flow network for sequence search. It models all pairwise similarities in $\hat{S}$ that belong to the first and last query image as source and sink, and tries to find an optimal flow between both. OPR [38] and VPR [39] are methods that extend this approach for more efficient descriptor comparisons and higher performance. Another graphical model is used in HMM [15], where the authors define a hidden markov model with emission and transition matrix for subsequent sequence search. SeqSLAM by Milford and Wyeth [22] searches for piecewise linear segments of high similarities in the similarity matrix $\hat{S}$. Similarly, ABLE [3] increases values in $\hat{S}$ that are part of a diagonal with high similarities. SMART [29] is an extension of SeqSLAM that additionally involves odometry for sequence search.

Different to the previously mentioned sequence-based approaches that operate on the similarity matrix $\hat{S}$, a few methods in the literature encode sequence information directly into postprocessed holistic descriptors. MCN [25] uses a neurologically inspired architecture to encode descriptor sequences of consecutive images into a new descriptor for each image. Garg et al. proposed Delta Descriptors [11] which uses a single, feature-wise convolution over all descriptors to encode sequence information in the output descriptors.

D. *Optimization via Iterated Conditional Modes (ICM)*

In our previous work [32], a modeled graph for place recognition is optimized via non-linear least squares optimization. While this approach outperforms state-of-the-art sequence-based methods, its optimization is relatively slow



Q
DB
s11
s12
s13
s21
s22
s23
s31
s32
s33
Q
DB
s11
s12
s13
s21
s22
s23
s31
s32
s33
Factors
unary:
$f_{prior}(s_{ij})$
binary:
$f^Q_{loop}(s_{ij},s_{kl}) + f^Q_{excl}(s_{ij},s_{kl})$
$f^{DB}_{loop}(s_{ij},s_{kl}) + f^{DB}_{excl}(s_{ij},s_{kl})$
n-ary:
$f_{seq}(s_{ij},s_{i-L/2,j-L/2},\dots,s_{i+L/2,j+L/2})$

Fig. 2. Illustration of the graph structure with nodes $s_{ij} \in S$ and factors f. See Sec. III for a detailed introduction to our used graphs.

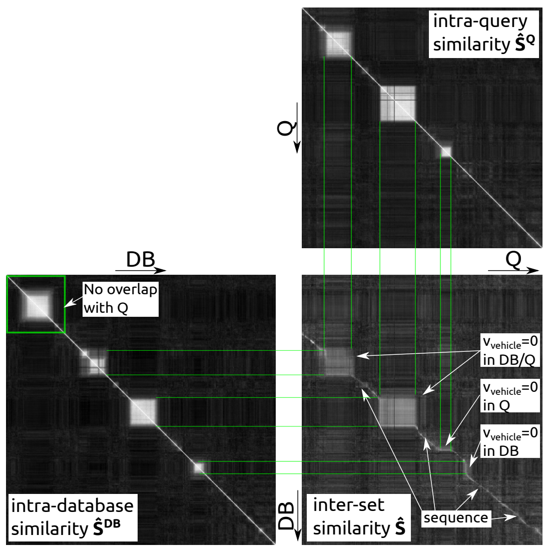


Fig. 3. An example with intra-database similarities $\hat{S}^{DB}$, intra-query similarities $\hat{S}^Q$ and inter-set similarities $\hat{S}$. The shape of high similarities in $\hat{S}^{DB}$ and $\hat{S}^Q$ affects the appearance of $\hat{S}$. Since the database and query images were recorded as spatio-temporal sequences, $\hat{S}$ contains a continuous trajectory (termed sequence) of high similarities. See Sec. III-A for further explanation.

and memory inefficient because of its need for the formulation and processing of a large Jacobian matrix J and a long vector of residuals r (cf. Sec. V). In [27, p.255], a more memory efficient non-linear least squares optimization is presented: Instead of using J and r for optimization, Gauss-Newton steps are performed on a matrix J^TJ and a vector J^Tr. However, this approach requires a successive calculation of J^TJ and J^Tr to be memory efficient which results in a potentially even longer runtime.

As discussed in the conclusion of [32], earlier work on graph optimization used hill-climbing techniques like ICM [17, p.599]. ICM (Iterated Conditional Modes) [6] optimizes each variable in a graph separately conditioned on the other variable's values. ICM has already been applied in robotics, for example in [12] for SLAM where ICM was used to optimize robot poses and landmark positions that were modeled in a Markov random field (MRF).

III. THE GRAPHICAL MODEL AT A GLANCE

The graphical model proposed in [32] was designed as a framework for place recognition. Different types of additional structural knowledge can be modeled in the graph. This combination of different knowledge allows the exploitation of all information in a single method instead of leveraging them in successive approaches. This joint exploitation of knowledge allows the inhibition of contradictory cues and the amplification of matching cues during graph optimization for better place recognition performance.

A factor graph [8] is used as graphical model that consists of nodes and factors (edges). For optimization, a quadratic cost function for each factor type is formulated that involves one or multiple nodes. The used factor types depend on the available additional structural knowledge. Two examples of a graph for place recognition are shown in Fig. 2.

A. Structural knowledge

Different types of additional structural knowledge can be leveraged in the graph. If a new type shall be used, a new factor with corresponding quadratic cost function has to be formulated. In [32] and in this paper, we exploit three types of additional structural knowledge:

1) Intra-database similarities $\hat{s}^{DB}_{ij} \in \hat{S}^{DB}$: $\hat{S}^{DB}$ contains pairwise descriptor similarities $\hat{s}^{DB}_{ij}$ between the i-th and j-th database image. Accordingly, $\hat{S}^{DB}$ is a symmetric matrix with maximum similarity along the main diagonal. Intra-database similarities are suited to find same places in the database due to stops and loops (high $\hat{s}^{DB}_{ij}$), but also to exclude them (low $\hat{s}^{DB}_{ij}$). This information is valuable because intra-set similarities affect the inter-set similarities $\hat{s}_{ij} \in \hat{S}$ (see Fig. 3) and reveal structures in $\hat{S}$.
2) Intra-query similarities $\hat{s}^Q_{ij} \in \hat{S}^Q$: The properties and advantages of intra-database similarities also apply for the pairwise intra-query similarities $\hat{s}^Q_{ij}$ (see Fig. 3).
3) Spatio-temporal sequences: If images in the database and in the query were recorded as spatio-temporal sequences, i.e. consecutive images i and $i+1$ are adjacent in the world, a trajectory of high similarities $\hat{s}_{ij}$ can be observed in $\hat{S}$ as can be seen in Fig. 3. This type of additional structural knowledge is widely used in the literature on place recognition (cf. Sec. II-C).

B. Nodes

The nodes $s_{ij} \in S$ model the similarities $\hat{s}_{ij} \in \hat{S}$ with $\hat{S} \in \mathbb{R}^{M\times N}$ from the pairwise descriptor comparisons between a set of M database images and a set of N query images. $\hat{s}_{ij}$ expresses the pairwise similarity between the i-th database image and the j-th query image. Each similarity $\hat{s}_{ij}$ is modeled as a separate node s_{ij} in the graph (see Fig. 2). The value of s_{ij} is modified during graph optimization.

C. Factors in the graph

Nodes in the graph are connected by different types of factors as shown in Fig. 2. The used factors depend on the available additional structural knowledge. Factors formulate dependencies between different nodes in the graph from this knowledge.

Factors and cost functions: In [32], we formulated the six different factors f_{prior}, f_{seq}, f^{DB}_{loop}, f^{Q}_{loop}, f^{DB}_{excl} and f^{Q}_{excl}:

- f_{prior} prevents too large deviations of node s_{ij} from its initial similarity $\hat{s}_{ij}$. It is the only obligatory factor that is used in every graph.
- f_{seq} is used in case of spatio-temporal sequences in database and query.
- f^{DB}_{loop} and f^{DB}_{excl} are used with intra-database similarities.
- f^{Q}_{loop} and f^{Q}_{excl} are used with intra-query similarities.

For each factor a corresponding quadratic cost function was formulated that involves one or more nodes and additional data. For example, the unary factor f_{prior} prevents too large deviations of each node s_{ij} from the initial similarities $\hat{s}_{ij}$ by ensuring

$$s_{ij} \approx \hat{s}_{ij} \tag{1}$$

with a corresponding quadratic cost function

$$f_{\text{prior}}(s_{ij}) = (s_{ij} - \hat{s}_{ij})^2 \tag{2}$$

The quadratic cost functions of factor f_{seq}, f^{DB}_{loop} and f^{Q}_{loop} express rules similar to Eq. (1) with cost functions similar to Eq. (2). Please refer to [32] for details about the expressed rules, corresponding cost functions, and their derivations. The factors f^{DB}_{excl} and f^{Q}_{excl} with their quadratic cost functions are different and discussed in the following Sec. IV.

D. Graph optimization

After graph creation, the graph has to be optimized. The optimization modifies the values of all nodes s_{ij} in order to get a refined version of $\hat{s}_{ij} \in \hat{S}$. The optimal values s^*_{ij} potentially better conform the dependencies and constraints that were introduced into the graph by the factors from additional knowledge.

Different optimization techniques can be employed. In [32] we used a non-linear least squares optimization (NLSQ). While NLSQ in [32] achieved very good performance, it was 1) relatively slow with $\sim$5.7 sec maximum runtime per query and 2) memory inefficient as it required a 500×500 patch-wise optimization of S due to memory limitations. See Sec. V for a more detailed discussion of runtime and memory limitations with NLSQ. In Sec. VI we propose an alternative optimization procedure based on Iterated Conditional Modes (ICM) that is much faster, much more memory efficient and even achieves better performance (cf. Sec. VIII-B).

IV. A NEW QUADRATIC COST FUNCTION FOR FACTOR f^{DB}_{EXCL} AND f^{Q}_{EXCL}

The rules and cost functions of the factors f^{DB}_{excl} and f^{Q}_{excl} are very different to the rules and cost functions of f_{prior}, f_{seq}, f^{DB}_{loop} and f^{Q}_{loop} that are all similar to Eq. (1) and (2) as already mentioned in Sec. III-C.

The rules of f^{DB}_{excl} and f^{Q}_{excl} that have to be expressed in a quadratic cost function for optimization are

$$f^{DB}_{\text{excl}}: \quad \neg(s_{ij}\uparrow \wedge s_{kj}\uparrow) \text{ iff } \hat{s}^{DB}_{ik}\downarrow \tag{3}$$

$$f^{Q}_{\text{excl}}: \quad \neg(s_{ij}\uparrow \wedge s_{il}\uparrow) \text{ iff } \hat{s}^{Q}_{jl}\downarrow \tag{4}$$

The rule of f^{DB}_{excl} expresses that "*if the intra-database similarity $\hat{s}^{DB}_{ik}$ between database image i and k is low ($\downarrow$), not both database images i and k can be similar to query image j at once.*" That means either s_{ij} or s_{kj} can be high ($\uparrow$) or neither if $\hat{s}^{DB}_{ik}$ is low ($\downarrow$). Please refer to [32] for a more detailed description.

A. The multiplication-based cost function from [32]

There are several ways to formulate a corresponding quadratic cost function for both rules in Eq. (3) and (4). In [32], the following multiplication-based cost functions were formulated:

$$f^{DB}_{\text{excl}} = (1 - \hat{s}^{DB}_{ik}) \cdot (s_{ij} \cdot s_{kj})^2 \tag{5}$$

$$f^{Q}_{\text{excl}} = (1 - \hat{s}^{Q}_{jl}) \cdot (s_{ij} \cdot s_{il})^2 \tag{6}$$

In Eq. (5), as long as s_{ij} is low, s_{kj} can be low or high without really increasing the cost, and vice versa. The factor $(1-\hat{s}^{DB}_{ik})$ ensures that this rule only applies if $\hat{s}^{DB}_{ik}$ is low ($\downarrow$). Same applies for f^{Q}_{excl} in Eq. (6).

B. The new minimum-based cost function

We propose alternative quadratic cost functions for the rules in Eq. (3) and (4) that are based on a minimum:

$$f^{DB}_{\text{excl}} = (1 - \hat{s}^{DB}_{ik}) \cdot \min(s_{ij}, s_{kj})^2 \tag{7}$$

$$f^{Q}_{\text{excl}} = (1 - \hat{s}^{Q}_{jl}) \cdot \min(s_{ij}, s_{il})^2 \tag{8}$$

In Eq. (7), the factor $(1 - \hat{s}^{DB}_{ik})$ again ensures that this rule only applies if $\hat{s}^{DB}_{ik}$ is low ($\downarrow$). As soon as s_{ij} is low, s_{kj} can be arbitrarily higher without affecting the cost, and vice versa. If both similarities s_{ij} and s_{kj} are high, the cost will be high as well. An advantage of the minimum-based cost functions could be the piecewise linearity, but a deeper analysis of this property is beyond the scope of this paper.

V. WHY NON-LINEAR LEAST SQUARES OPTIMIZATION OF THE GRAPH IS INEFFICIENT

After the formulation of a graph (Sec. III), the nodes s_{ij} have to be optimized in order to get an improved similarity matrix S out of $\hat{S}$. In [32] a graph was optimized using a non-linear least squares optimization procedure. It was implemented in Python with scipy's *least_squares*-optimization function and the *Trust Region Reflective* algorithm.

The used implementation requires the formulation of a Jacobian matrix J and a vector of residuals r. The size of J depends on the total number of factors $\#f$ and the number of nodes #nodes with

$$J \in \mathbb{R}^{\#f \times \#\text{nodes}} \tag{9}$$

Note that J is sparse since only two nodes are involved in most cost functions. The size of vector r solely depends on $\#f$ with

$$r \in \mathbb{R}^{\#f} \tag{10}$$

The number of nodes (#nodes) in a graph depends on the number of database images M and the number of query images N with

$$\#\text{nodes} = M \cdot N \tag{11}$$

According to the way the different factor types are used [32], the total number of factors $\#f$ can be quite high:

$$\#f = \underbrace{2 \cdot M \cdot N}_{\#f_{\text{prior}} + \#f_{\text{seq}}} + \underbrace{2 \cdot N \cdot \binom{M}{2}}_{\#f^{DB}_{\text{loop}} + \#f^{DB}_{\text{excl}}} + \underbrace{2 \cdot M \cdot \binom{N}{2}}_{\#f^{Q}_{\text{loop}} + \#f^{Q}_{\text{excl}}} \tag{12}$$

Operations for optimization over the potentially extremely high number of elements in J and r cause a high computational effort. Moreover, due to the nonlinearity of many factors, J (like r) has to be updated after each optimization step. This leads to long runtimes.

The cubically increasing number of factors leads to an extremely high memory usage even for smaller datasets. For a relatively small dataset with 1000 database and 1000 query images, the Jacobian matrix J would have $(2 \cdot 10^9) \times 10^6$ elements. Even if J is represented as a sparse matrix with approx. two entries per row and 4 Bytes per entry, the memory usage would be $>$14.9 GB; the corresponding vector of residuals r would require approx. 7.5 GB of memory.

There are optimization methods for non-linear least squares optimization that reduce the memory requirements by operating on a matrix J^TJ and a vector J^Tr ([27, p.255], cf. Sec. II-D). But these would probably even increase the runtime as a tradeoff, because the elements of J^TJ and J^Tr have to be computed successively in order to keep the memory consumption low. A detailed investigation of this optimization approach is beyond the scope of this paper and part of future work. To partially circumvent the high memory usage of J and r, we performed in [32] a partial optimization of S on approx. (500×500)-sized patches.

In Sec. VI, we propose a much faster and memory efficient optimization based on ICM. It fully avoids the allocation and processing of huge matrices like J and r. And as we will show in Sec. VIII-B, the ICM-based graph optimization for place recognition even outperforms the already good results from [32] as it can optimize the full S-matrix instead of patches.

VI. GRAPH OPTIMIZATION WITH ICM

In the previous Sec. V, we gave an intuition why the non-linear least squares optimization (NLSQ) used in [32] is relatively slow and memory inefficient. We figured out two main reasons: 1) the allocation and updating of the potentially huge Jacobian matrix J and the vector of residuals r and 2) the operation on J and r for optimization. For a much faster and memory efficient graph optimization, we propose the usage of the alternative optimization scheme ICM that fully avoids the usage of huge matrices like J and r.

A. The basic idea of ICM

ICM (Iterated Conditional Modes) was proposed by Besag for the optimization of Markov Random Fields for image denoising [6]. The basic idea of ICM is as follows: Instead of minimizing an error function $E(\forall s_{ij} \in S)$ (e.g., the sum over all factor's cost functions) to optimize all nodes at once, each node is optimized separately conditioned on all other nodes by using their values from the previous iteration. This tremendously reduces the complexity of the error function $E(\cdot)$ and simplifies its minimization. This procedure is repeated several iterations until convergence or for a fixed number of iterations. The application of ICM for the optimization of the graph for place recognition in Sec. VI-B below will clarify this idea.

B. ICM-based graph optimization for place recognition

In order to optimize the graph with its nodes $s_{ij} \in S$, the global error function $E(\cdot)$ has to be minimized with

$$s^*_{ij} \in S^* = \underset{s_{ij} \in S}{\arg\min}\, E(\forall s_{ij} \in S) \tag{13}$$

$E(\cdot)$ is a function of the $M \cdot N$ nodes $s_{ij} \in S$. It is the sum over all quadratic cost functions $f(\cdot)$ of all factors in the graph:

$$E(\forall s_{ij} \in S) = \sum_{\forall f \in Graph} f(\cdot) \tag{14}$$

With ICM, we define $E(\cdot)$ to be a set of error functions $E_{ij}(s_{ij} \mid \cdot)$. Each $E_{ij}(s_{ij} \mid \cdot)$ is a function of a single node s_{ij} conditioned on the other nodes. $E_{ij}(s_{ij} \mid \cdot)$ is the sum over all quadratic cost functions f that depend on s_{ij}:

$$E_{ij}(s_{ij} \mid \cdot) = \sum_{\forall f(s_{ij} \mid \cdot) \in Graph} f(s_{ij} \mid \cdot) \tag{15}$$

Accordingly, each error function $E_{ij}(s_{ij} \mid \cdot)$ is a *quadratic function* of only a single node s_{ij}. This allows a conversion into the standard form

$$E(s_{ij} \mid \cdot) = a \cdot s^2_{ij} + b \cdot s_{ij} + c \tag{16}$$

with coefficients a, b and c. Given this standard form, the optimal value s^*_{ij} can be easily computed with

$$s^*_{ij} = \underset{s_{ij}}{\arg\min}(E(s_{ij} \mid \cdot)) = -\frac{b}{2a} \tag{17}$$

c can be neglected for computational efficiency. For fast computation, all s^*_{ij} should be computed synchronously based on S from the previous iteration (S_{t-1}) as already proposed in [6].

TABLE I
PARTIAL COEFFICIENTS a_f AND b_f FOR EACH FACTOR WITH CORRESPONDING COST FUNCTION.

$\boldsymbol{f(s_{ij})}$	$\boldsymbol{a_f}$	$\boldsymbol{b_f}$
	$f_{\text{prior}} = (s_{ij} - \hat{s}_{ij})^2$	
f_{prior}	1	$-2 \cdot \hat{s}_{ij}$
	$f_{\text{loop}}^{DB} = \hat{s}_{ik}^{DB}(s_{ij} - s_{kj})^2$; $f_{\text{loop}}^{Q} = \hat{s}_{jl}^{Q}(s_{ij} - s_{il})^2$	
f_{loop}^{DB}	$\frac{w_{\text{loop}}^{DB}}{M-1} \cdot \sum_{\forall k \setminus i} \hat{s}_{ik}^{DB}$	$-\frac{2 \cdot w_{\text{loop}}^{DB}}{M-1} \cdot \sum_{\forall k \setminus i} \hat{s}_{ik}^{DB} s_{kj}$
f_{loop}^{Q}	$\frac{w_{\text{loop}}^{Q}}{N-1} \cdot \sum_{\forall l \setminus j} \hat{s}_{jl}^{Q}$	$-\frac{2 \cdot w_{\text{loop}}^{Q}}{N-1} \cdot \sum_{\forall l \setminus j} \hat{s}_{jl}^{Q} s_{il}$
	$f_{\text{excl}}^{DB} = (1 - \hat{s}_{ik}^{DB})(s_{ij} \cdot s_{kj})^2$; $f_{\text{excl}}^{Q} = (1 - \hat{s}_{jl}^{Q})(s_{ij} \cdot s_{il})^2$	
f_{excl}^{DB}	$\frac{w_{\text{excl}}^{DB}}{M-1} \cdot \sum_{\forall k \setminus i}(1 - \hat{s}_{ik}^{DB}) \cdot s_{kj}^2$	0
f_{excl}^{Q}	$\frac{w_{\text{excl}}^{Q}}{N-1} \cdot \sum_{\forall l \setminus j}(1 - \hat{s}_{jl}^{Q}) \cdot s_{il}^2$	0
	$f_{\text{excl}}^{DB} = (1 - \hat{s}_{ik}^{DB}) \min(s_{ij}, s_{kj})^2$, $f_{\text{excl}}^{Q} = (1 - \hat{s}_{jl}^{Q}) \min(s_{ij}, s_{il})^2$	
f_{excl}^{DB}	$\frac{w_{\text{excl}}^{DB}}{M-1} \cdot \sum_{\forall k \mid s_{ij} < s_{kj}} (1 - \hat{s}_{ik}^{DB})$	0
f_{excl}^{Q}	$\frac{w_{\text{excl}}^{Q}}{N-1} \cdot \sum_{\forall l \mid s_{ij} < s_{il}} (1 - \hat{s}_{jl}^{Q})$	0
	$f_{\text{seq}} = (s_{ij} - \bar{s}_{ij})^2$	
f_{seq}	w_{seq}	$-2 \cdot w_{\text{seq}} \cdot \bar{s}_{ij}$

C. *The computation of a and b*

a and b depend on the used factors and their quadratic cost functions, and in turn on the used structural knowledge. They can be easily computed with

$$a = \sum_{\forall f} a_f, \quad b = \sum_{\forall f} b_f \tag{18}$$

a_f and b_f are the partial coefficients of each factor type. The coefficients for the factors used in this paper are listed in Table I. They were derived from the factor's quadratic cost functions that were converted into the standard form in Eq. (16). Note that a normalization term $1/(M{-}1)$ or $1/(N{-}1)$ was added for f_{loop}^{DB}, f_{loop}^{Q}, f_{excl}^{DB} and f_{excl}^{Q}.

For example, if we want to use intra-database similarities $\hat{S}^{DB}$ together with the *multiplicative* cost function f_{excl}^{DB}, we simply have to sum up the coefficients a_f and b_f for f_{prior}, f_{loop}^{DB} and f_{excl}^{DB} from Table I, and insert them into Eq. (17) to obtain the equation for optimal values s_{ij}^*:

$$s_{ij}^* = \frac{\hat{s}_{ij} + \frac{w_{\text{loop}}^{DB}}{M-1} \cdot \sum_{\forall k \setminus i} \hat{s}_{ik}^{DB} s_{kj}}{1 + \frac{w_{\text{loop}}^{DB}}{M-1} \cdot \sum_{\forall k \setminus i} \hat{s}_{ik}^{DB} + \frac{w_{\text{excl}}^{DB}}{M-1} \cdot \sum_{\forall k \setminus i}(1 - \hat{s}_{ik}^{DB}) \cdot s_{kj}^2} \tag{19}$$

D. *The full ICM-based graph optimization procedure for place recognition*

The full algorithmic approach of the ICM-based graph optimization for place recognition is shown in Algorithm 1. It includes a matrix normalization (Line 1-3), node initialization (Line 4), the application of a sequence-based method (see Sec. VII) in case of sequence exploitation (Line 7), the ICM-based optimization itself (Lines 8-11) and a check for convergence (Lines 16-17).

During our experiments, we noticed a possible divergence of s_{ij} in case of a too high weighting w_{seq} of the sequence information. Therefore, we added a simple strategy to our ICM-based optimization procedure (Line 12-15): If any s_{ij} starts to diverge, w_{seq} is simply reduced by 0.1 and the optimization is repeated.

Algorithm 1: ICM-based graph optimization procedure

```
Data: $\hat{s}_{ij} \in \hat{S}$, $\hat{s}_{ij}^{DB} \in \hat{S}^{DB}$, $\hat{s}_{ij}^{Q} \in \hat{S}^{Q}$
Input: parameters $w_{\text{loop}}^{DB}$, $w_{\text{excl}}^{DB}$, $w_{\text{loop}}^{Q}$, $w_{\text{excl}}^{Q}$, $w_{\text{seq}}$
Result: $s_{ij} \in S$ with $S \in \mathbb{R}^{M \times N}$
   // normalize all similarities to $\hat{s}_{ij}, \hat{s}_{ij}^{DB}, \hat{s}_{ij}^{Q} \in [0,1]$
1  $\hat{S} := norm(\hat{S})$
2  $\hat{S}^{DB} := norm(\hat{S}^{DB})$
3  $\hat{S}^{Q} := norm(\hat{S}^{Q})$
   // initialize nodes $s_{ij} \in S$
4  $S := \hat{S}$
5  $S_{t-1} := \hat{S}$
   // optimize $S$ with ICM
6  for $iteration = 1, \ldots, \#iterations$ do
       // compute all $\bar{s}_{ij} \in \bar{S}$ with seqConv (Sec. VII)
7      $\forall i,j : \bar{s}_{ij} = \text{SeqConv}(\{ij\}; s_{kl} \in S_{t-1})$
       // optimize all variables $s_{ij} \in S$
8      for $\forall i,j$ do
9          $a := \sum_{\forall f} a_f(\{i,j\} | S_{t-1}, \hat{S}, \hat{S}^{DB}, \hat{S}^{Q}, \bar{S})$
10         $b := \sum_{\forall f} b_f(\{i,j\} | S_{t-1}, \hat{S}, \hat{S}^{DB}, \hat{S}^{Q}, \bar{S})$
11         $s_{ij} := -\frac{b}{2a}$
       // check for divergence
12     if $\max |S_{t-1} - S| > 2$ and $w_{seq} > 0$ then
           // decrease sequence weight $w_{\text{seq}}$
13         $w_{\text{seq}} = \max(w_{\text{seq}} - 0.1, 0)$
           // initialize nodes $s_{ij} \in S$
14         $S := \hat{S}$
15         $S_{t-1} := \hat{S}$
       // check for convergence
16     if $\max |S_{t-1} - S| < 1e-4$ then
           // stop optimization if converged
17         break
       // store optimization result for next iteration
18     $S_{t-1} := S$
   // return optimized similarities $s_{ij} \in S$
19 return $S$
```

VII. THE SEQUENCE-BASED METHOD SEQCONV

The quadratic cost function of factor $f_{\text{seq}}(s_{ij})$ (cf. Sec. III-C) with

$$f_{\text{seq}}(s_{ij}) = w_{seq} \cdot (s_{ij} - \bar{s}_{ij})^2 \tag{20}$$

requires a sequence-based method, similar to those in Sec. II-C, which computes the maximum average similarity $\bar{s}_{ij}$ of L similarities s_{kl} around s_{ij}. This method 1) has to be fast to not slow down the graph optimization process, 2) must be memory efficient to not increase the memory requirements of the graph optimization and 3) should achieve as good performance as possible to boost the performance of the graph-based approach for place recognition.

For this purpose, we propose *SeqConv*, a simple and thus fast and memory efficient sequence-based method. It can be formulated in a single equation:

$$\begin{aligned}\bar{s}_{ij} &= \text{SeqConv}(\{ij\}, s_{kl} \in S) \\ &= \max_{v \in \{v_{min}..v_{max}\}} \frac{1}{L} \sum_{l=j-\lfloor L/2 \rfloor}^{j+\lfloor L/2 \rfloor} s_{i+round(v \cdot l),l} \end{aligned} \quad (21)$$

SeqConv tries to find a line segment with length L in S around s_{ij}. At the borders of S, the line segment is truncated and L is decreased. v defines the slope of the line in S; in our work, v equals $\{v_{min}..v_{max}\} = \{0.8, 0.9, 1, 1.1, 1.2\}$.

SeqConv can be efficiently implemented using $|\{v_{min}..v_{max}\}|$ convolutions. Each slope v can be expressed as a separate convolutional kernel. This formulation as convolution allows the usage of highly optimized software libraries or even GPU-based implementations like tensorflow.

In our experimental results in Sec. VIII, we will demonstrate that SeqConv is fast, memory efficient, and achieves high performance compared to sequence-based approaches from the literature.

VIII. Experimental Results

In the algorithmic design of our ICM-based graph optimization for place recognition (short ICM), we were targeting on the creation of an algorithm that 1) at least maintains the performance of the non-linear least squares based graph optimization for place recognition (short NLSQ) from [32] while being 2) fast and 3) memory efficient.

In the following experiments, we will show that the performance of NLSQ is not only maintained but partially improved with ICM (Sec. VIII-B). The performance of the new minimum-based cost function for f_{excl} is investigated in Sec. VIII-C. Moreover, we will show that the full setup of ICM with sequence exploitation significantly outperforms six state-of-the-art sequence-based methods (Sec. VIII-D), present corresponding runtimes and memory usages (Sec. VIII-E) and demonstrate the versatility of ICM on four different holistic image descriptors from the literature (Sec. VIII-F).

A. Experimental Setup

1) Image descriptors: Like in [32], *NetVLAD* [2] is used as CNN-image descriptor throughout all experiments including runtime and memory measurements. For *NetVLAD*, we use the author's implementation trained on the Pitts30k dataset with VGG-16 and whitening. In Sec. VIII-F, we evaluate performances with three additional holistic image descriptors: 1) *AlexNet* [35]: the flattened conv3-layer trained on ImageNet from Matlab is used, 2) *DenseVLAD* [36]: the author's implementation with provided weights is used and 3) *HybridNet* [7]: the flattened conv5-layer from the author's implementation with provided weights is used.

2) Metric: The performance is measured with average precision which is the area under the precision-recall curve.

3) Datasets: We use the same datasets as described in [32]: All experiments are based on the five different datasets Nordland [34], StLucia (Various Times of the Day) [14], CMU (Visual Localization) [4], GardensPoint (Walking) [13] and Oxford (RobotCar) [20]. For Oxford, we sampled different sequences with one frame per second and use the high accuracy position data from [21] as ground truth.

4) Implementation: For the NLSQ-based graph optimization, we use our Python implementation from [32]. Our ICM-based graph optimization is implemented in Matlab.

5) Parameters: For the NLSQ-based graph optimization, we use the same parameters as in [32] with $w^{DB}_{\text{loop}}=w^{DB}_{\text{loop}}=1$, $w^{DB}_{\text{excl}}=w^{DB}_{\text{excl}}=20$ and $w_{\text{seq}}=10$. With ICM, we only changed w_{seq} to 0.5 for ICM$_{mul}$ or 0.2 for ICM$_{min}$. ICM$_{mul}$ uses the multiplicative (Sec. IV-A) and ICM$_{min}$ the minimum-based cost function f_{excl} (Sec. IV-B). At most 200 iterations for optimization with ICM were performed. While NLSQ optimizes at most (500×500)-patches of S at once (like in [32]) due to memory limitations, ICM performs optimization always on the full graph. For all sequence-based methods, we set the sequence length $L = 11$; other parameters are set as proposed in the corresponding papers.

B. Optimization with NLSQ vs ICM$_{mul}$

In a first experiment, we compare the performance of the NLSQ-based optimization with the ICM$_{mul}$-based optimization. The three used graphs are identical to the graphs in [32] and model different amounts of additional knowledge. Results are show in Table II.

Both graph optimizations outperform the raw NetVLAD descriptor. In comparison, NLSQ and ICM$_{mul}$ perform similarly with additional structural knowledge from $\hat{S}^{DB}$ and $\hat{S}^{DB}+\hat{S}^{Q}$. Given sequence information ($+Seq$), ICM$_{mul}$ outperforms NLSQ by 10% on average, presumably because ICM$_{mul}$ optimizes the full graph at once while a patch-wise optimization has to be done with NLSQ and because of a better control of the optimization procedure. The full optimization allows a more widespread usage of structural knowledge even between distant similarities in S. The results demonstrate the superiority of ICM$_{mul}$ over NLSQ: ICM$_{mul}$ achieves better performance and allows a full optimization of S due to its memory efficiency.

C. Comparison of the multiplicative and the minimum-based cost function f_{excl} (ICM$_{mul}$ vs ICM$_{min}$)

Table II compares the performances of the ICM-based graph optimization with the multiplication-based (Sec. IV-A) and the minimum-based (Sec. IV-B) cost functions f^{DB}_{excl} and f^{Q}_{excl}. ICM$_{min}$ slightly outperforms ICM$_{mul}$, particularly with additional information from $\hat{S}^{DB}+\hat{S}^{Q}$. The low performance gap indicates that both cost functions can be used interchangeably.

D. Comparison of ICM with state-of-the-art sequence-based methods

The full setup of ICM exploits the same sequence information as sequence-based methods from the literature. To show

TABLE II

AVERAGE PRECISION WITH NETVLAD FOR DIFFERENT CONFIGURATIONS AND GRAPH OPTIMIZATION APPROACHES. COLORED ARROWS INDICATE LARGE ($\geq$25% BETTER/ WORSE) OR MEDIUM ($\geq$10%) DEVIATION COMPARED TO THE RAW DESCRIPTOR PERFORMANCE. BOLD TEXT INDICATES THE BEST PERFORMANCE PER ROW AND PER INTRA-DATABASE SOURCE.

				Additional Structural Knowledge								
				$\hat{S}^{DB}$			$\hat{S}^{DB} + \hat{S}^{Q}$			$\hat{S}^{DB} + \hat{S}^{Q} + Seq$		
Dataset	**Database**	**Query**	**Raw** [2]	**NLSQ** [32]	**ICM**$_{mul}$ (ours)	**ICM**$_{min}$ (ours)	**NLSQ** [32]	**ICM**$_{mul}$ (ours)	**ICM**$_{min}$ (ours)	**NLSQ** [32]	**ICM**$_{mul}$ (ours)	**ICM**$_{min}$ (ours)
Nordland	fall	spring	0.39	0.50 ↑	0.50 ↑	0.58 ↑	0.52 ↑	0.52 ↑	0.63 ↑	0.93 ↑	0.98 ↑	**1.00** ↑
	fall	winter	0.06	0.09 ↑	0.09 ↑	0.14 ↑	0.14 ↑	0.14 ↑	0.21 ↑	0.42 ↑	0.83 ↑	**0.98** ↑
	spring	winter	0.11	0.16 ↑	0.16 ↑	0.17 ↑	0.24 ↑	0.24 ↑	0.30 ↑	0.60 ↑	0.92 ↑	**0.99** ↑
	summer	spring	0.32	0.45 ↑	0.45 ↑	0.54 ↑	0.48 ↑	0.48 ↑	0.59 ↑	0.92 ↑	0.97 ↑	**1.00** ↑
	summer	fall	0.63	0.74 ↗	0.75 ↗	0.82 ↑	0.77 ↗	0.77 ↗	0.87 ↑	**1.00** ↑	0.88 ↑	**1.00** ↑
StLucia	100909-0845	190809-0845	0.41	0.46 ↗	0.46 ↗	0.51 ↗	0.50 ↗	0.50 ↗	0.57 ↑	0.74 ↑	0.80 ↑	**0.87** ↑
	100909-1000	210809-1000	0.47	0.52 ↗	0.52 ↗	0.56 ↗	0.55 ↗	0.55 ↗	0.61 ↑	0.80 ↑	0.84 ↑	**0.88** ↑
	100909-1210	210809-1210	0.51	0.56 →	0.56 →	0.58 ↗	0.60 ↗	0.59 ↗	0.62 ↗	0.86 ↑	0.87 ↑	**0.89** ↑
	100909-1410	190809-1410	0.38	0.46 ↗	0.46 ↗	0.49 ↑	0.49 ↑	0.49 ↑	0.54 ↑	0.79 ↑	0.85 ↑	**0.90** ↑
	110909-1545	180809-1545	0.27	0.33 ↗	0.32 ↗	0.42 ↑	0.34 ↑	0.35 ↑	0.48 ↑	0.49 ↑	0.71 ↑	**0.90** ↑
CMU	20110421	20100901	0.73	0.74 →	0.74 →	0.72 →	0.75 →	0.73 →	0.74 →	0.81 ↗	**0.85** ↗	0.83 ↗
	20110421	20100915	0.77	0.78 →	0.78 →	0.76 →	0.77 →	0.77 →	0.77 →	**0.85** ↗	**0.85** ↗	0.84 →
	20110421	20101221	0.56	0.58 →	0.58 →	0.53 →	0.59 →	0.58 →	0.54 →	**0.65** ↗	0.64 ↗	0.64 ↗
	20110421	20110202	0.61	0.67 →	0.67 →	0.63 →	0.69 ↗	0.67 →	0.66 →	0.83 ↑	**0.87** ↑	0.80 ↑
GardensPoint	day-right	day-left	0.97	0.98 →	0.98 →	**1.00** →	0.98 →	0.98 →	**1.00** →	**1.00** →	**1.00** →	**1.00** →
	day-right	night-right	0.46	0.50 →	0.50 →	0.53 ↗	0.56 ↗	0.56 ↗	0.74 ↑	0.82 ↑	0.98 ↑	**1.00** ↑
	day-left	night-right	0.34	0.38 ↗	0.38 ↗	0.34 →	0.43 ↑	0.43 ↑	0.46 ↑	0.78 ↑	0.95 ↑	**1.00** ↑
Oxford	2014-12-09-13-21-02	2015-05-19-14-06-38	0.78	0.92 ↗	0.87 ↗	0.85 →	0.89 ↗	0.85 →	0.88 ↗	0.92 ↗	**0.95** ↗	0.91 ↗
	2014-12-09-13-21-02	2015-08-28-09-50-22	0.60	0.69 ↗	0.70 ↗	0.66 →	0.71 ↗	0.66 →	0.71 ↗	0.68 ↗	**0.77** ↑	0.73 ↗
	2014-12-09-13-21-02	2014-11-25-09-18-32	0.87	0.89 →	**0.90** →	**0.90** →	**0.90** →	**0.90** →	0.89 →	0.88 →	0.87 →	0.85 →
	2014-12-09-13-21-02	2014-12-16-18-44-24	0.55	0.54 →	0.58 →	0.46 ↘	0.65 ↗	0.62 ↗	0.54 →	0.77 ↑	**0.85** ↑	0.70 ↑
worst			0.06	0.09	0.09	0.14	0.14	0.14	0.21	0.42	0.64	0.64
best			0.97	0.98	0.98	1.00	0.98	0.98	1.00	1.00	1.00	1.00
average			0.51	0.57	0.57	0.58	0.60	0.59	0.64	0.79	0.87	0.89

TABLE III

AVERAGE PRECISION WITH NETVLAD FOR THE COMPARISON OF OUR ICM-BASED METHODS WITH SEQUENCE-BASED METHODS FROM THE LITERATURE.

				Additional Structural Knowledge								
				$\hat{S}^{DB} + \hat{S}^{Q} + Seq$		Seq						
Dataset	**Database**	**Query**	**Raw** [2]	**ICM**$_{mul}$ (ours)	**ICM**$_{min}$ (ours)	**SeqConv** (ours)	**MCN** [25]	**ABLE** [3]	**VPR** [39]	**OPR** [38]	**SeqSLAM** [22]	**Delta** [11]
Nordland	fall	spring	0.39	0.98 ↑	**1.00** ↑	0.95 ↑	0.54 ↑	0.95 ↑	0.68 ↑	0.90 ↑	0.89 ↑	0.39 →
	fall	winter	0.06	0.83 ↑	**0.98** ↑	0.63 ↑	0.20 ↑	0.44 ↑	0.26 ↑	0.19 ↑	0.72 ↑	0.16 ↑
	spring	winter	0.11	0.92 ↑	**0.99** ↑	0.80 ↑	0.26 ↑	0.66 ↑	0.39 ↑	0.18 ↑	0.83 ↑	0.23 ↑
	summer	spring	0.32	0.97 ↑	**1.00** ↑	0.95 ↑	0.43 ↑	0.94 ↑	0.64 ↑	0.73 ↑	0.91 ↑	0.41 ↑
	summer	fall	0.63	0.88 ↑	**1.00** ↑	**1.00** ↑	0.47 ↓	**1.00** ↑	0.89 ↑	0.97 ↑	0.95 ↑	0.51 ↘
StLucia	100909-0845	190809-0845	0.41	0.80 ↑	**0.87** ↑	0.76 ↑	0.56 ↑	0.51 ↗	0.47 ↗	0.54 ↑	0.12 ↓	0.73 ↑
	100909-1000	210809-1000	0.47	0.84 ↑	**0.88** ↑	0.84 ↑	0.60 ↑	0.60 ↑	0.48 →	0.55 ↗	0.12 ↓	0.74 ↑
	100909-1210	210809-1210	0.51	0.87 ↑	**0.89** ↑	0.83 ↑	0.65 ↑	0.61 ↗	0.47 →	0.55 →	0.14 ↓	0.74 ↑
	100909-1410	190809-1410	0.38	0.85 ↑	**0.90** ↑	0.81 ↑	0.51 ↑	0.57 ↑	0.42 ↗	0.53 ↑	0.14 ↓	0.75 ↑
	110909-1545	180809-1545	0.27	0.71 ↑	**0.90** ↑	0.55 ↑	0.39 ↑	0.36 ↑	0.41 ↑	0.52 ↑	0.13 ↓	0.64 ↑
CMU	20110421	20100901	0.73	0.81 ↗	**0.83** ↗	0.81 ↗	0.81 ↗	0.76 →	0.47 ↓	0.49 ↓	0.03 ↓	0.36 ↓
	20110421	20100915	0.77	**0.85** ↗	0.84 →	**0.85** ↗	0.79 →	0.77 →	0.47 ↓	0.50 ↓	0.07 ↓	0.54 ↓
	20110421	20101221	0.56	**0.64** ↗	**0.64** ↗	**0.64** ↗	0.61 →	0.59 →	0.43 ↘	0.47 ↘	0.05 ↓	0.32 ↓
	20110421	20110202	0.61	**0.87** ↑	0.80 ↑	0.75 ↗	0.80 ↑	0.65 →	0.47 ↘	0.45 ↓	0.13 ↓	0.51 ↘
GardensPoint	day-right	day-left	0.97	**1.00** →	**1.00** →	**1.00** →	0.98 →	**1.00** →	0.69 ↓	0.69 ↓	0.42 ↓	**1.00** →
	day-right	night-right	0.46	0.98 ↑	**1.00** ↑	0.74 ↑	0.47 →	0.68 ↑	0.52 ↗	0.64 ↑	0.34 ↓	0.80 ↑
	day-left	night-right	0.34	0.95 ↑	**1.00** ↑	0.74 ↑	0.36 →	0.63 ↑	0.35 →	0.49 ↑	0.13 ↓	0.64 ↑
Oxford	2014-12-09-13-21-02	2015-05-19-14-06-38	0.78	**0.95** ↗	0.91 ↗	0.79 →	0.92 ↗	0.64 ↘	0.54 ↓	0.01 ↓	0.06 ↓	0.37 ↓
	2014-12-09-13-21-02	2015-08-28-09-50-22	0.60	**0.77** ↑	0.73 ↗	0.56 →	0.49 ↘	0.38 ↓	0.37 ↓	0.01 ↓	0.05 ↓	0.29 ↓
	2014-12-09-13-21-02	2014-11-25-09-18-32	**0.87**	**0.87** →	0.85 →	0.81 →	0.75 ↘	0.65 ↓	0.60 ↓	0.02 ↓	0.07 ↓	0.44 ↓
	2014-12-09-13-21-02	2014-12-16-18-44-24	0.55	**0.85** ↑	0.70 ↑	0.68 ↗	0.51 →	0.56 →	0.17 ↓	0.09 ↓	0.01 ↓	0.24 ↓
worst			0.06	**0.64**	**0.64**	0.55	0.20	0.36	0.17	0.01	0.01	0.16
best			0.97	**1.00**	**1.00**	**1.00**	0.98	**1.00**	0.89	0.97	0.95	**1.00**
average			0.51	0.87	**0.89**	0.79	0.58	0.66	0.49	0.45	0.30	0.51

the superiority of ICM, we compared it with the six sequence-based approaches MCN [25], ABLE [3], VPR [39], OPR [38], SeqSLAM [22] and Delta (Descriptors) [11]. In addition, we also evaluate our proposed method SeqConv (Sec. VII) which is used in the factors $f_{\text{seq}}(\cdot)$.

Table III shows the corresponding results. Both ICM-based approaches achieve best performance on most datasets. Particularly on average, our ICM-methods perform 32% to 190% better than the sequence-based approaches from the literature. While VPR, OPR and SeqSLAM potentially fail by design in case of loops and stops in the database but could benefit in the absence of loops and stops, MCN, ABLE and Delta are able to deal with loops and stops. Nevertheless, all methods from the literature fail to beat the performance of both ICM-based approaches on a single dataset, even on datasets without loops and stops (Nordland, GardensPoint).

SeqConv as an algorithmic part of ICM contributes to ICM's performance. The results in Table III demonstrate that SeqConv alone already outperforms on average all methods from the literature. However, ICM even exceeds the performance of SeqConv. This demonstrates the superiority of fusing and exploiting all available structural knowledge at once in a single graph with ICM.

E. Runtime and memory usage

For all evaluated methods in Table II and III, we measured the maximum runtime per query and the maximum memory usage. Results are shown in Table IV. Note that NLSQ could not be applied to full graphs but only to subgraphs due to memory limitations, while ICM optimized full graphs – the memory usage of NLSQ on full graphs would be much higher.

A target of this paper is the design of a fast and memory efficient graph optimization for place recognition. The runtimes and memory usages of NLSQ, ICM$_{mul}$ and ICM$_{min}$ in Table IV demonstrate that we could actually tremendously speed up the graph optimization and clearly reduce the required memory. With $\hat{S}^{DB}$, ICM$_{mul}$ was almost $2800\times$ faster than NLSQ and required approx. $280\times$ less memory. In case of the full setup with sequence exploitation ($+Seq$), ICM$_{mul}$ was approx. $385\times$ faster than NLSQ and required more than $60\times$ less memory. The current implementation of ICM$_{min}$ is much slower than ICM$_{mul}$, but still $2.5\times$ to $10\times$ faster than NLSQ and $60\times$ to $235\times$ more memory efficient.

Compared to the sequence-based methods, ICM$_{mul}$ achieves a comparable runtime and memory usage. In summary, a maximum runtime of $9.1\,\mathrm{msec}$ and memory usage of 490MB with ICM$_{mul}$ is perfectly suited for many real-time applications.

The results in Table IV also show that SeqConv as an algorithmic part of ICM is extremely fast and very memory efficient. It required only $95\,\mu$sec per query, which is faster than most of the evaluated approaches. Its 170MB memory usage is comparable to the other sequence-based methods from the literature.

F. Performance with three additional descriptors

In a final experiment, we want to show the versatility of our ICM-based graph optimization approach for place recognition. Therefore, we repeated the experiments from Sec. VIII-D on the 21 datasets with three additional image descriptors. The same parameters as before were used. Table V shows the obtained performances.

Both ICM-based methods again clearly outperform on average all other approaches over all descriptors. The results show that our proposed methods perform well no matter which descriptor is used, because the optimization operates only on the descriptor similarities but not directly on the descriptors.

IX. Conclusion

In this paper, we presented a novel approach for fast and memory efficient place recognition that achieves superior performance compared to [32] and six state-of-the-art sequence-based methods from the literature.

Our graph optimization with ICM could increase the maximum performance of [32] with NLSQ on average by 10% while being $385\times$ faster and $60\times$ more memory efficient. Specifically, the *maximum* runtime with ICM on our largest dataset was 9.1msec per query which is totally suited for real-time applications. Further, we could show that ICM with sequence exploitation achieves superior performance compared to six state-of-the-art sequence-based methods from the literature: It outperformed them on NetVLAD by *at least* 32% on average over 21 datasets with comparable runtime and memory consumption. No sequence-based method could outperform both ICM-based methods on a single dataset.

ICM was also applied on three additional image descriptors without parameter adjustment and significantly outperformed all other methods. Adapted parameters that better fit the statistics of each descriptor would presumably further improve the performance.

An interesting question for future work is the application of the graph optimization with ICM for place recognition on sparse similarity matrices $\hat{S}$ that are returned by methods like [33, 38] for efficient place recognition on large datasets. Another interesting direction is the extension of the graph-based framework for additional structural knowledge like odometry. Odometry for example could be modeled either directly with additional factors or indirectly by replacing SeqConv with an alternative like SMART [29] that leverages odometry during sequence search.

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6): 1437–1451, 2018.

[3] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera. Towards life-long visual localization using an efficient matching of binary sequences from images. In *International Conference on Robotics and Automation (ICRA)*, 2015.

[4] H. Badino, D. Huber, and T. Kanade. Visual topometric localization. In *Intelligent Vehicles Symposium (IV)*, 2011.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision (ECCV)*, 2006.

[6] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986.

[7] Z. Chen, A. Jacobson, N. Sünderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford. Deep learning features at scale for visual place recognition. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[8] Frank Dellaert and Michael Kaess. Factor graphs for robot perception. *Foundations and Trends in Robotics*, 6 (1-2):1–139, 2017.

[9] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-Net: A trainable CNN for joint description and detection of local features. In

TABLE IV

RUNTIMES AND MEMORY USAGES WITH NETVLAD FOR ALL GRAPH-BASED METHODS COMPARED TO THE SEQUENCE-BASED METHODS FROM THE LITERATURE. THE MAXIMUM RUNTIME PER QUERY ON THE BIGGEST DATASET IS SHOWN (DATASET OXFORD 2014-12-09-13-21-02 – 2014-11-25-09-18-32 WITH $M = 2133$ DATABASE IMAGES AND $N = 2253$ QUERY IMAGES). ALL RUNTIMES WERE MEASURED WITH AN INTEL I7-7700K CPU WITH 64GB RAM. (*MEMORY USAGE FOR THE OPTIMIZATION OF A SUBGRAPH ACCORDING TO SEC. VIII-A5)

	Graph-based methods								
	$\hat{S}^{DB}$			$\hat{S}^{DB}+\hat{S}^{Q}$			$\hat{S}^{DB}+\hat{S}^{Q}+Seq$		
Method	**NLSQ** [32]	**ICM**$_{mul}$ (ours)	**ICM**$_{min}$ (ours)	**NLSQ** [32]	**ICM**$_{mul}$ (ours)	**ICM**$_{min}$ (ours)	**NLSQ** [32]	**ICM**$_{mul}$ (ours)	**ICM**$_{min}$ (ours)
max runtime per query	5.3sec	1.9msec	521msec	2.4sec	1.5msec	1sec	3.5sec	9.1msec	1.1sec
max memory usage	60.5GB*	220MB	262MB	30.4GB*	292MB	300MB	30.5GB*	490MB	512MB

	Sequence-based methods						
Method	**SeqConv** (ours)	**MCN** [25]	**ABLE** [3]	**VPR** [39]	**OPR** [38]	**SeqSLAM** [22]	**Delta** [11]
max runtime per query	95μsec	247msec	43μsec	3.8msec	3.4msec	6.3msec	431μsec
max memory usage	170MB	586MB	114MB	162MB	125MB	234MB	140MB

TABLE V

AVERAGE PRECISION WITH FOUR DIFFERENT DESCRIPTORS. THE RESULTS SHOW THE CONCLUSION OF EXPERIMENTS OVER THE 21 DATASETS THAT WERE ALSO USED IN TABLE II AND III.

			Additional Structural Knowledge								
			$\hat{S}^{DB}+\hat{S}^{Q}+Seq$					Seq			
Descriptor	**Case**	**Raw**	**ICM**$_{mul}$ (ours)	**ICM**$_{min}$ (ours)	**SeqConv** (ours)	**MCN** [25]	**ABLE** [3]	**VPR** [39]	**OPR** [38]	**SeqSLAM** [22]	**Delta** [11]
NetVLAD [2]	worst	0.06	**0.64**	**0.64**	0.55	0.20	0.36	0.17	0.01	0.01	0.16
	best	0.97	**1.00**	**1.00**	**1.00**	0.98	**1.00**	0.89	0.97	0.95	**1.00**
	average	0.51	0.87	**0.89**	0.79	0.58	0.66	0.49	0.45	0.30	0.51
AlexNet [35]	worst	0.07	0.12	0.19	0.06	**0.21**	0.05	0.17	0.01	0.03	0.02
	best	0.94	**1.00**	**1.00**	**1.00**	0.98	**1.00**	0.99	**1.00**	0.97	0.82
	average	0.50	**0.72**	**0.72**	0.63	0.66	0.54	0.55	0.53	0.31	0.48
DenseVLAD [36]	worst	0.09	**0.64**	0.33	0.16	0.14	0.12	0.15	0.01	0.03	0.12
	best	0.96	**1.00**	**1.00**	**1.00**	0.91	**1.00**	0.95	0.98	0.95	**1.00**
	average	0.57	0.87	**0.88**	0.76	0.62	0.66	0.49	0.34	0.31	0.53
HybridNet [7]	worst	0.08	0.15	0.17	0.09	**0.22**	0.05	0.19	0.01	0.03	0.07
	best	0.94	**1.00**	**1.00**	**1.00**	0.98	**1.00**	0.98	**1.00**	0.97	0.90
	average	0.54	0.76	**0.79**	0.68	0.69	0.59	0.56	0.56	0.31	0.56

Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[10] Paul Furgale and Timothy D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560, 2010.

[11] S. Garg, B. Harwood, G. Anand, and M. Milford. Delta Descriptors: Change-based place representation for robust visual localization. *IEEE Robotics and Automation Letters (RA-L)*, 5(4):5120–5127, 2020.

[12] Javier Gimenez, Adriana Amicarelli, Juan Toibero, Fernando Sciascio, and Ricardo Carelli. Continuous probabilistic SLAM solved via iterated conditional modes. *International Journal of Automation and Computing*, 16 (6):838–850, 2019.

[13] Arren Glover. Day and night with lateral pose change datasets, 2014.

[14] Arren Glover, Will Maddern, Michael Milford, and Gordon Wyeth. FAB-MAP + RatSLAM: Appearance-based SLAM for Multiple Times of Day. In *International Conference on Robotics and Automation (ICRA)*, 2010.

[15] P. Hansen and B. Browning. Visual place recognition using HMM sequence matching. In *Int. Conf. on Intel. Robots and Systems*, 2014.

[16] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[17] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 2012.

[19] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics (T-RO)*, 32(1):1–19, 2016.

[20] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The Oxford RobotCar dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[21] Will Maddern, Geoffrey Pascoe, Matthew Gadd, Dan Barnes, Brian Yeomans, and Paul Newman. Real-time Kinematic Ground Truth for the Oxford RobotCar Dataset. *arXiv preprint: 2002.10152*, 2020.

[22] Michael Milford and Gordon Fraser Wyeth. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *International Conference on Robotics and Automation (ICRA)*, 2012.

[23] Tayyab Naseer, Luciano Spinello, Wolfram Burgard, and Cyrill Stachniss. Robust visual robot localization across seasons using network flows. In *Conference on Artificial Intelligence*, 2014.

[24] Peer Neubert and Stefan Schubert. Hyperdimensional computing as a framework for systematic aggregation of image descriptors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[25] Peer Neubert, Stefan Schubert, and Peter Protzel. A neurologically inspired sequence processing model for mobile robot place recognition. *IEEE Robotics and Automation Letters (RA-L)*, 4(4), 2019.

[26] Peer Neubert, Stefan Schubert, and Peter Protzel. Resolving place recognition inconsistencies using intra-set similarities. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):2084–2090, 2021.

[27] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*, chapter Least-Squares Problems, pages 245–269. Springer New York, 2006.

[28] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large-scale image retrieval with attentive deep local features. In *International Conference on Computer Vision (ICCV)*, 2017.

[29] E. Pepperell, P. I. Corke, and M. J. Milford. All-environment visual place recognition with SMART. In *International Conference on Robotics and Automation (ICRA)*, 2014.

[30] Stefan Schubert and Peer Neubert. What makes visual place recognition easy or hard? *CoRR*, abs/2106.12671, 2021.

[31] Stefan Schubert, Peer Neubert, and Peter Protzel. Unsupervised learning methods for visual place recognition in discretely and continuously changing environments. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[32] Stefan Schubert, Peer Neubert, and Peter Protzel. Graph-based non-linear least squares optimization for visual place recognition in changing environments. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):811–818, 2021.

[33] Stefan Schubert, Peer Neubert, and Peter Protzel. Beyond ANN: Exploiting structural knowledge for efficient place recognition. In *International Conference on Robotics and Automation (ICRA)*, 2021.

[34] Niko Sünderhauf, Peer Neubert, and Peter Protzel. Are we there yet? Challenging SeqSLAM on a 3000 km journey across all four seasons. *International Conference on Robotics and Automation (ICRA) Workshop on Long-Term Autonomy*, 2013.

[35] Niko Sünderhauf, Sareh Shirazi, Feras Dayoub, Ben Upcroft, and Michael Milford. On the performance of convnet features for place recognition. In *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[36] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[37] C. Valgren and A. J. Lilienthal. SIFT, SURF and seasons: Long-term outdoor localization using local features. In *European Conference on Mobile Robots (ECMR)*, 2007.

[38] O. Vysotska and C. Stachniss. Lazy data association for image sequences matching under substantial appearance changes. *IEEE Robotics and Automation Letters (RA-L)*, 1(1):213–220, 2016.

[39] O. Vysotska and C. Stachniss. Relocalization under substantial appearance changes using hashing. *International Conference on Intelligent Robots and Systems (IROS) Workshop PPNIV'17*, 2017.

[40] O. Vysotska, T. Naseer, L. Spinello, W. Burgard, and C. Stachniss. Efficient and effective matching of image sequences under substantial appearance changes exploiting gps priors. In *International Conference on Robotics and Automation (ICRA)*, 2015.